МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка Android-приложения для озвучивания книг

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-345.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.
_____ Б.А. Марков

Автор работы,
студент группы КЭ-403
_____ К.С. Мицукова

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студентке группы КЭ-403
Мицуковой Ксении Станиславовне,
обучающейся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)

Разработка Android-приложения для озвучивания книг.

2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.

3. **Исходные данные к работе**

3.1. Синицын И.В. Разработка мобильных приложений: учебное пособие. / И.В. Синицын, Е.А. Чернов, Ю.А. Воронцов. // Москва: РТУ МИРЭА, 2023. – 162 с.

3.2. Android Studio: среда разработки мобильных приложений. [Электронный ресурс] URL: https://arduinoplus.ru/android-studio (дата обращения: 29.01.2024 г.).

3.3. Kotlin для Android: теперь официально. [Электронный ресурс] URL: https://habr.com/ru/company/JetBrains/blog/329028 (дата обращения: 29.01.2024 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Выполнить анализ предметной области и произвести обзор существующих решений.

4.2. Спроектировать архитектуру системы.

4.3. Реализовать программное обеспечение для озвучивания книг.

4.4. Провести тестирование мобильного приложения.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н. Б.А. Марков

Задание принял к исполнению К.С. Мицукова

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

В последние годы активно развиваются мобильные приложения. Стремительное распространение связано с комфортным использованием интернет-технологий благодаря смартфонам. С годами использование смартфонов только увеличивается, и теперь в ежедневный обиход активно входит использование телефона, что обусловлено многофункциональностью современных аппаратов. Сейчас количество возможностей мобильных устройств растет и позволяет использовать их для самых разных целей: от покупки одежды до заказа такси, и все это не без помощи мобильных приложений.

Мобильные приложения – это программное обеспечение, разрабатываемое для определенных платформ и предназначенное для работы на мобильных устройствах. Они могут выполнять как развлекательные функции, так и использоваться для рабочих процессов. Актуальность разработки мобильных приложений обусловлена спросом потребителей на мобильные устройства и доходами, получаемыми от продаж самих приложений, платных подписок или дополнительных услуг [1].

На данный момент область электронных книг стремительно развивается. В 2023 году объем книг, выпускаемых исключительно в формате электронной книги, увеличился на 22% по сравнению с предыдущим годом, а их продажи – на 16% [2]. Эти данные показывают, что современный формат выпуска книг является удобным для пользователей, соответственно, приложения для чтения книг и прослушивания их в аудиоформате являются актуальными. Они позволяют пользователю легко приобрести желаемую книгу и иметь ее всегда под рукой вне зависимости от местонахождения.

Постановка задачи

Целью выпускной квалификационной работы является разработка Android-приложения для озвучивания книг.

Для достижения поставленной цели необходимо решить следующие задачи:

1) выполнить анализ предметной области и произвести обзор существующих решений;
2) спроектировать архитектуру системы;
3) реализовать программное обеспечение для озвучивания книг;
4) провести тестирование мобильного приложения.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 57 страниц, объем списка литературы – 24 источника.

В первой главе был проведен анализ предметной области, определены задачи разрабатываемого приложения, выполнен обзор аналогичных приложений, а также произведен обзор существующих решений.

Во второй главе описаны функциональные и нефункциональные требований к приложению, варианты использования мобильного приложения, а также описана его архитектура.

В третьей главе описаны подробности реализации приложения: средства реализации, диаграмма классов, реализация локальной базы данных, описана реализация компонентов приложения, пользовательского интерфейса, а также подробно описан алгоритм загрузки книги в приложение.

В четвертой главе описан процесс тестирования разработанного приложения и представлены полученные результаты тестирования.

В заключении приводятся основные результаты и рассматриваются дальнейшие направления исследования.

В приложении приведены листинги программы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области мобильного приложения

Книги с древних времен являются неотъемлемой частью жизни людей. Они используются не только в качестве источника знаний, но и как инструмент для передачи культурного наследия от одного поколения к другому. Книги до сих пор остаются одним из главных средств обучения, развития и развлечения человека.

С развитием технологий книги претерпели множество изменений. Например, появилась новая форма книг – интерактивные книги. Популярным примером интерактивных книг являются детские издания и книжки-раскладушки. Однако в последнее время под термином «интерактивная книга» подразумевают объединение текстового содержания книги с мультимедийными элементами: видео, анимационные материалы, звуковые эффекты и задания в игровой форме. Интерактивные книги призваны сделать процесс чтения более увлекательным и простым, а также предоставить читателю возможность взаимодействовать с содержанием книги, например, направлять персонажа или менять сюжет.

В настоящее время термин «интерактивная книга» тесно связан с термином «электронная книга». Электронные книги – это книги, которые доступны в цифровом формате и могут быть прочитаны на различных устройствах, таких как компьютеры, планшеты и смартфоны. Электронные книги приобрели популярность и элементы интерактивных книг – гипертекстовые ссылки, анимацию, видео и звук [3].

Помимо интерактивных книг популярным форматом чтения стали аудиокниги. Аудиокниги позволяют людям наслаждаться литературой, даже если у них нет времени или возможности читать печатные книги, например, по пути на работу.

Современные технологии позволяют создавать аудио версии книг с помощью автоматической озвучки. В данном подходе используется искусственный интеллект для синтеза речи. Нейросетевые технологии позволяют

озвучить книги без участия человека, что помогает расширить доступность удобного аудиоформата книг для широкого круга читателей [4].

1.2. Обзор существующих аналогов мобильного приложения

На данный момент существуют различные приложения, предоставляющие пользователям возможность прочтения книг в формате интерактивных книг, включая в себя книги в аудиоформате.

В качестве аналогов рассматриваются следующие приложения: «Букмейт», «Литрес», «MyBook» и «Читалка».

Букмейт [5]

Данное приложение является одним из самых популярных в категории приложений для чтения книг и прослушивания их в аудиоформате. Скриншоты данного приложения представлены на рисунке 1.

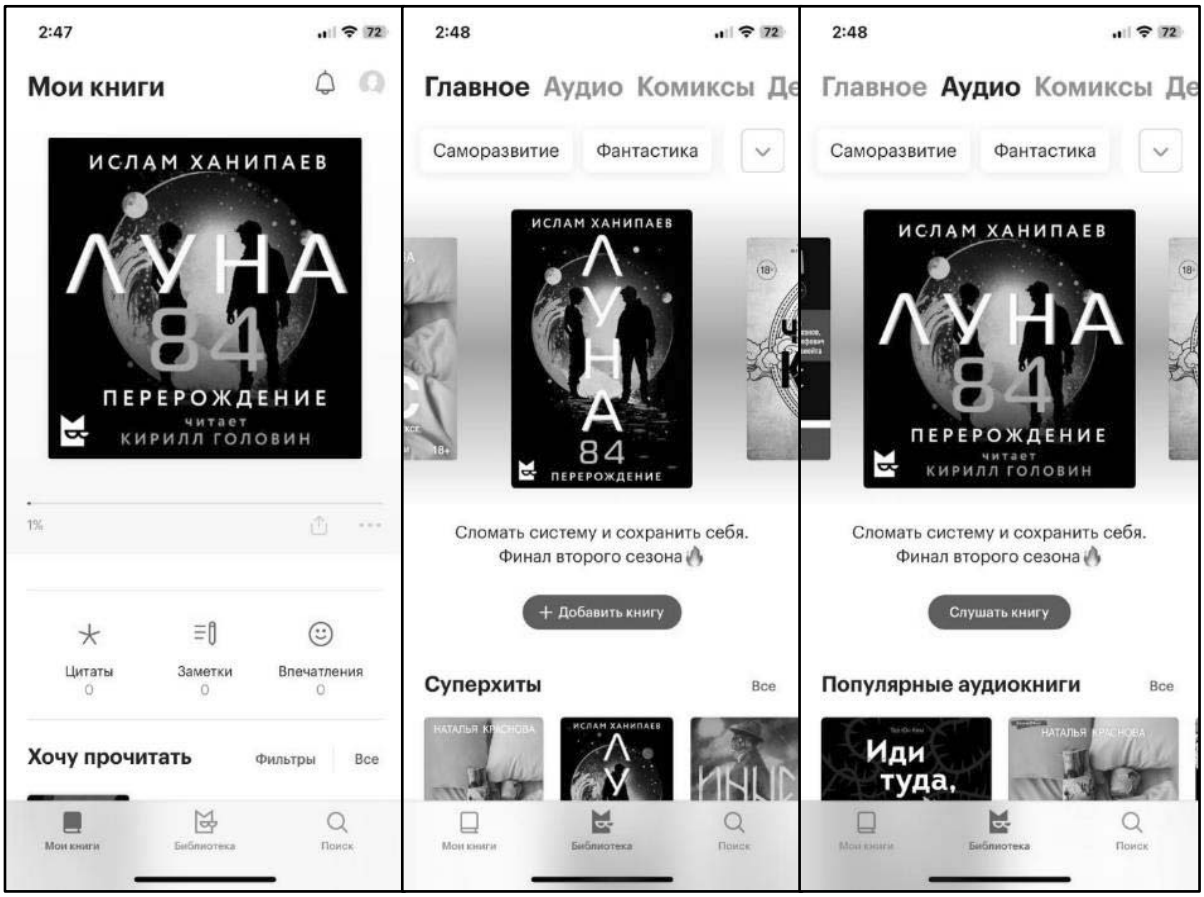


Рисунок 1 – Скриншоты приложения для чтения книг Букмейт

Приложение Букмейт предназначено для чтения электронных книг интерактивного формата и является кроссплатформенным. Данное приложение предлагает широкий выбор книг различных жанров и авторов, которые доступны для скачивания и чтения в оффлайн режиме.

Литрес [6]

Данное приложение так же предназначено для чтения и прослушивания электронных книг. Скриншоты приложения Литрес представлены на рисунке 2.

Рисунок 2 – Скриншоты приложения для чтения книг Литрес

С помощью приложения Литрес пользователи могут приобретать и скачивать книги из библиотеки, включающей в себя произведения различных жанров и авторов. Также приложение предлагает рекомендации для прочтения на основе прочитанных пользователем книг.

MyBook [7]

Рассматриваемое приложение представляет собой магазин, в котором можно приобрести книги в электронном формате.

Скриншоты приложения MyBook представлены на рисунке 3.

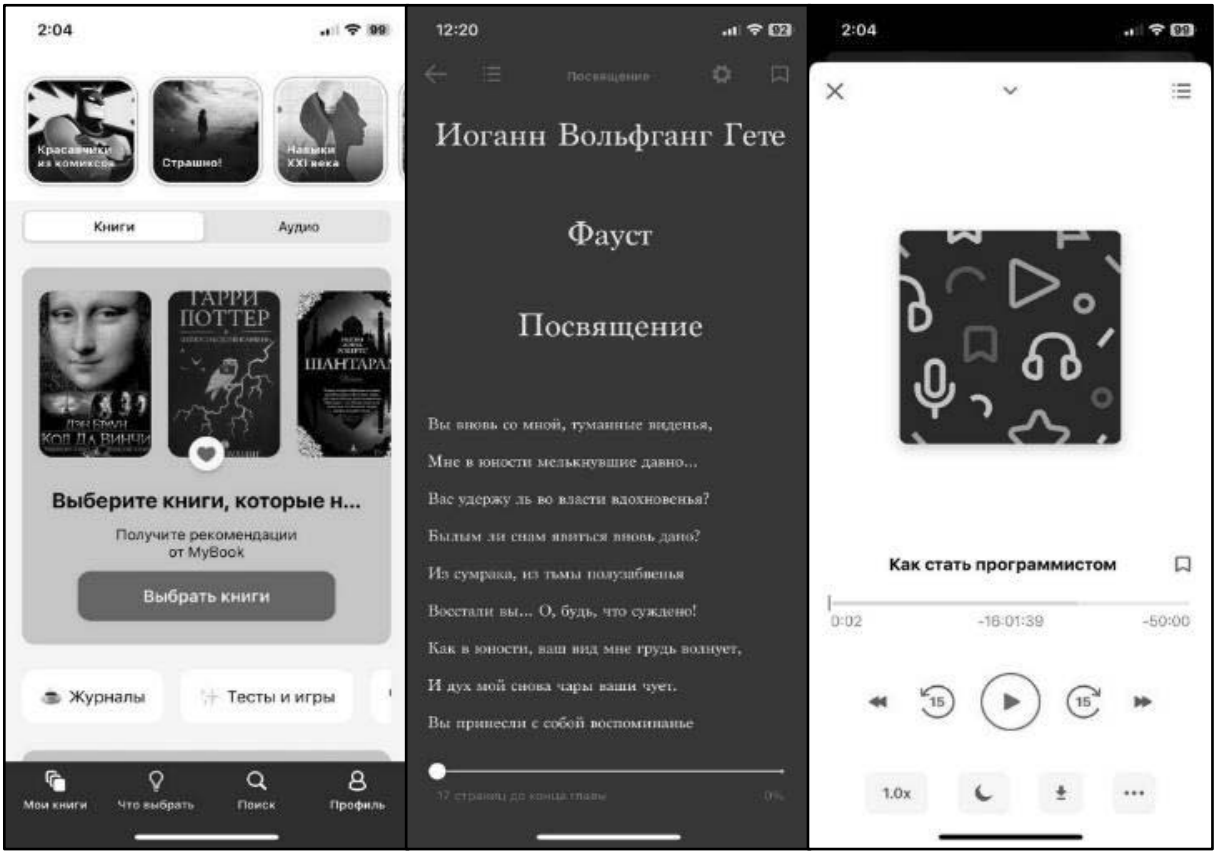


Рисунок 3 – Скриншоты приложения для чтения книг MyBook

Приложение MyBook является кроссплатформенным и предоставляет пользователям возможность скачивать и читать книги в различных форматах. Также приложение предлагает персонализированные рекомендации книг на основе предпочтений пользователя и возможность делиться книгами с друзьями.

Читалка (в приложении Яндекс Старт) [8]

Данное приложение является интегрированным приложением для чтения электронных книг, доступное на платформе Яндекс Старт.

На рисунке 4 представлены скриншоты приложения для чтения книг Читалка.

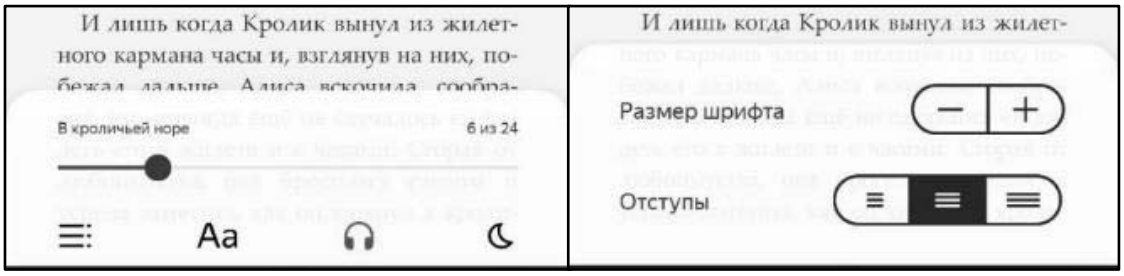


Рисунок 4 – Скриншоты приложения для чтения книг Читалка

Приложение Читалка предоставляет пользователю доступ к библиотеке книг различных жанров и авторов, а также возможность загружать книги с устройства и управлять ими. Данное приложение включает в себя функции для настройки внешнего вида текста и возможность чтения книг в оффлайн режиме.

1.3. Сравнение аналогов разрабатываемого приложения

Рассмотрим отличия обозреваемых аналогов разрабатываемого приложения и определим основной функционал.

Основные функциональные блоки приложения Букмейт:

- поиск книг в библиотеке приложения;
- фильтрация по категориям;
- система рекомендаций книг;
- добавление книг в раздел сохраненных книг;
- сохранение прогресса прочтения книг;
- загрузка книг в приложение;
- сохранение цитат;
- прослушивание книг в аудиоформате;
- настройка скорости озвучивания книг;
- личный кабинет пользователя.

Приложение Литрес имеет схожий функционал с приложением Букмейт, однако есть и отличия: имеются новости в формате историй и отсутствует возможность загрузки пользователем книг с устройства.

В приложении MyBook имеются те же функциональные блоки, за исключением возможности загрузки пользователем книг с устройства. Также стоит отметить то, что в данном приложении, в отличие от остальных обозреваемых аналогов, на главной странице расположены сборники книг, объединенные тематикой, например, похожие на произведение «Мастер и Маргарита», однако пользователю не предоставляется возможность просмотреть полный список книг, не входящих в сборники.

Функционал приложения Читалка в сравнении с предыдущими ограничен: настройки скорости прослушивания менее вариативны, а также нельзя настроить таймер для отключения аудиокниги. Однако, приложение Читалка наиболее схоже с реализуемым приложением, так как имеет функцию озвучивания книги с помощью нейросетевых технологий.

Выделим основные функциональные блоки разрабатываемого приложения:

1) загрузка книг в приложение;
2) удаление книги;
3) фильтрация по алфавиту;
4) добавление книг в раздел избранных книг;
5) просмотр списка избранных книг;
6) удаление книг из раздела избранных книг;
7) просмотр списка книг;
8) изменение отображения списка книг;
9) просмотр содержимого книги;
10) сохранение прогресса прочтения книг;
11) сохранение цитат;
12) просмотр списка сохраненных цитат;
13) удаление цитат;

14) настройка оформления книги;

15) поиск по словам в книге;

16) прослушивание книг в аудиоформате с озвучкой с помощью нейросетевых технологий;

17) настройка скорости озвучивания книг;

18) настройка голоса озвучки.

Таким образом, в ходе проведенного обзора аналогов приложений были выявлены основные функциональные блоки разрабатываемого приложения для озвучивания книг.

1.4. Обзор существующих технологий для реализации проекта

Платформы мобильных устройств и языки программирования

В мире мобильных устройств две операционные системы, Android и iOS, выделяются как наиболее популярные, охватывая более 80% всех активных смартфонов.

Рассмотрим операционную систему Android, разработанную Google на базе ядра Linux. Данная операционная система является открытой платформой, предоставляющей пользователям большую свободу в настройке устройства. Для продвинутых пользователей требование ручной настройки мобильного устройства является отличной возможностью подстроить смартфон под свои нужды.

Операционная система Android сильно уступает в аспекте безопасности устройства, так как платформа не защищает неопытных пользователей от вирусов, и даже программы-антивирусы не всегда способны в этом помочь. Однако открытость платформы Android позволяет пользователям скачивать из интернета нелицензированные версии приложений из интернета.

Рассмотрим далее языки программирования, на которых реализуются приложения для операционной системы Android.

Для написания приложений для Android используются такие языки программирования, как Java и Kotlin. Рассмотрим каждый из них.

Объектно-ориентированный язык программирования Java, разрабатываемый компанией компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года, является одним из самых широко применяемых языков программирования на протяжениии многих лет [9]. Программы, написанные на Java, транслируются в байт-код, который выполняется на JVM.

JVM – Java Virtual Machine. Основная часть исполняющей системы Java, так называемой Java Runtime Environment. Виртуальная машина Java исполняет байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java (javac) [10].

Язык Java обладает преимуществами, которые делают его предпочтительным выбором для многих разработчиков. Простой и знакомый синтаксис Java облегчает начало работы с языком, а надежная и удобная в использовании среда разработки предоставляет программистам мощные инструменты для создания приложений. Благодаря этим характеристикам, Java пользуется широкой популярностью среди разработчиков разного уровня опыта, позволяя им эффективно и быстро реализовывать свои идеи в виде новых программ и приложений.

Язык программирования Java долгое время был основным для разработки приложений под Android, однако, начиная с 2017 года, Google официально поддерживает Kotlin, признанный за его безопасность и совместимость с Java [11].

Рассмотрим язык программирования Kotlin. Данный язык программирования позволяет создавать как объектно-ориентированный код с основными его принципами (использование классов, наследования и полиморфизма), что делает его похожим на язык Java, так и функциональный код. Также Kotlin является языком программирования со статической типизацией, что означает, что проверка типов выполняется во время компиляции, а также что выполнять действия с переменной неподходящего типа невозможно. Язык программирования Kotlin полностью совместим с Java, и это

позволяет использовать в одной программе одновременно оба языка при необходимости.

В сравнении с Java язык программирования Kotlin обладает меньшей скоростью компиляции: по статистике Java на 12–15 быстрее Kotlin при сборке [12].

Среды разработки мобильных приложений

Рассмотрим среды разработки мобильных приложений, так как правильный выбор IDE является важнейшим этапом в разработке.

IDE (Integrated Development Environment) – это виды программного обеспечения, предназначенные для работы над приложениями, их разработки и тестирования [13].

Eclipse, IntelliJ IDEA и Android Studio являются известными IDE, которые предлагают разработчикам разнообразные инструменты для удобного создания и отладки приложений. Каждая из них имеет свои преимущества и недостатки, что позволяет выбрать наиболее подходящую среду в зависимости от потребностей и предпочтений разработчика. Рассмотрим каждую среду разработки подробнее.

Eclipse – это интегрированная среда разработки с широким функционалом для разработчиков программного обеспечения. Eclipse поддерживает множество языков программирования, однако наибольшую популярность она обрела у Java-разработчиков [14].

Выделим преимущества данной IDE:

1) пошаговая сборка кода, позволяющая сразу же проверить измененные места кода на ошибки;

2) наличие функции объединения нескольких связанных проектов в единую рабочую область, в которой можно задать общие настройки для них;

3) наличие инструмента для контроля утечек памяти.

Также определим недостатки Eclipse:

1) низкая производительность;

2) сложность работы в связи с отсутствием интуитивного интерфейса: эффективно использовать данную среду разработки может только опытный пользователь.

IntelliJ IDEA – умная среда от известной компании JetBrains предназначена для разработки на языках программирования Java и Kotlin [15].

Преимущества данной среды разработки:

1) наличие инструментов для работы с базами данных;
2) инструменты для запуска тестов и анализа покрытия кода;
3) удобная навигация в IDE.

В качестве недостатка можно выделить то, что разработчику придется потратить время на то, чтобы разобраться с инструментами данной среды разработки.

Android Studio – программа, являющаяся средой разработки приложений для мобильной платформы Android [16].

Основные особенности Android Studio:

1) удобный редактор;
2) быстрый и встроенный эмулятор, который можно настраивать как угодно, позволяющий просмотреть приблизительные показатели производительности при запуске на настоящем устройстве;
3) поддержка всех устройств с операционной системой Android;
4) встроенная поддержка различных сервисов от компании Google.

Выводы по первой главе

В результате анализа предметной области были определены задачи разрабатываемого в рамках выпускной квалификационной работы приложения. Обзор аналогичных решений позволил определить основной функционал приложений для чтения книг. Обзор существующих решений позволил рассмотреть существующие среды разработки и языки программирования для разработки Android-приложения, в результате чего было принято решение разработать мобильное приложение на платформе Android с использованием языка программирования Kotlin в среде разработки Android Studio.

2. ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

2.1. Требования к мобильному приложению

Функциональные требования

При разработке программного обеспечения функциональные требования определяют функции, которые должно выполнять все приложение или только один из его компонентов, то есть функциональные требования описывают, что должна делать система [17].

Рассмотрим функциональные требования системы. Система Audible должна обеспечивать доступ к следующим возможностям.

1. Загрузка книг в приложение.
2. Удаление книг.
3. Фильтрация книг по алфавиту.
4. Добавление книг в раздел избранных.
5. Просмотр списка избранных книг.
6. Удаление книг из раздел избранных.
7. Просмотр списка книг.
8. Изменение отображения списка книг.
9. Просмотр содержимого книги.
10. Сохранение прогресса прочтения книг.
11. Сохранение цитат.
12. Просмотр списка сохраненных цитат.
13. Удаление цитат.
14. Настройка оформления книг.
15. Поиск по словам в книгах.
16. Прослушивание книг в аудиоформате с озвучкой с помощью нейросетевых технологий.
17. Настройка скорости озвучивания книг.
18. Настройка голоса озвучки.

Нефункциональные требования

Нефункциональные требования определяют стандарты производительности и атрибуты качества программного обеспечения, например, удобство использования системы, эффективность, безопасность, масштабируемость. То есть функциональные требования определяют, что система делает, а нефункциональные – как система это делает [17].

Система Audible должна соответствовать следующим требованиям.

1. Быть разработанной для операционной системы Android.
2. Быть разработанной на языке программирования Kotlin.
3. Корректная работа на системах Android версии 4.1 и выше.

2.2. Варианты использования приложения

На основе требований, предъявляемых к разрабатываемому приложению, была разработана диаграмма вариантов использования [18], которая изображена на рисунке 5.

Рисунок 5 – Варианты использования приложения

В проектируемом приложении представлен один актер – пользователь. Пользователем является любой человек, который запустил приложение.

Пользователь может загрузить книгу в приложение с помощью файлового менеджера устройства.

Пользователь может отфильтровать загруженные книги в приложение по алфавиту.

Пользователь может изменить отображение списка книг с «плитки» на структурированный список, в котором отображаются название книги и автор.

Пользователь может просмотреть список загруженных книг. При этом пользователь может добавить книгу в избранное, удалить книгу из приложения, а также открыть книгу, нажав на ее обложку. При открытии книги пользователь может сохранить цитату, а также просмотреть список цитат. При просмотре списка цитат пользователь может удалить цитату. Помимо этого, при открытии книги пользователь может выполнить поиск по словам в книге, настроить оформление книги и прослушать книгу в аудиоформате. При прослушивании книги в аудиоформате пользователь может настроить скорость озвучивания книги и голос озвучки.

Пользователь может просмотреть список добавленных в избранное книг. При этом пользователь может удалить книгу из избранного.

2.3. Архитектура мобильного приложения

Из широкого выбора подходов в реализации архитектур мобильных приложений можно выбрать популярный архитектурный паттерн Model-View-ViewModel, у которого имеются следующие преимущества:

1) разделение логики приложения и пользовательского интерфейса;
2) чистый и легко поддерживаемый код;
3) улучшенная масштабируемость.

Благодаря данным преимуществам был сделан выбор в пользу данного подхода при разработке мобильного приложения для озвучивания книг. Архитектурный подход MVVM изображен на рисунке 6.

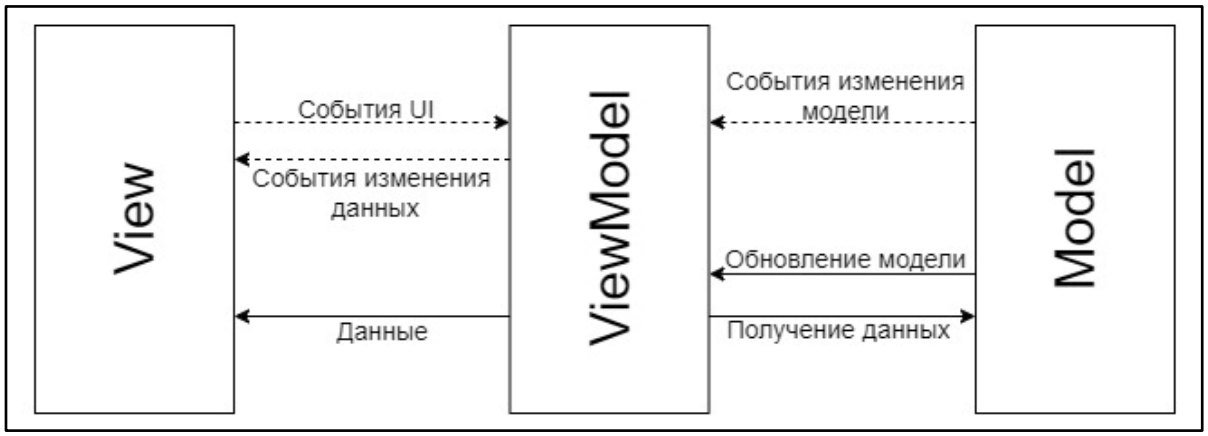


Рисунок 6 – Схема архитектурного подхода MVVM

MVVM состоит из трех компонентов: модель (Model), представление (View) и ViewModel. Модель содержит данные и бизнес-логику приложение, представление отображает данные пользователю, а ViewModel является прослойкой между предыдущими компонентами: обрабатывает данные, введенные пользователем, и обновляет модель.

Рассмотрим принцип работы архитектурного подхода MVVM.

1. Пользователь выполняет какое-либо действие на пользовательском интерфейсе.

2. Представление (View) обрабатывает действие и отправляет запрос на ViewModel.

3. ViewModel обрабатывает запрос, используя данные из модели (Model) и возвращает результаты представлению.

4. Представление обновляется на основе результатов ViewModel.

5. Если изменения произошли в ViewModel, он обновляет модель.

6. Если изменения произошли в модели, ViewModel получает уведомление об изменении и обновляет представление [19].

Диаграмма компонентов разрабатываемого приложения представлена на рисунке 7.

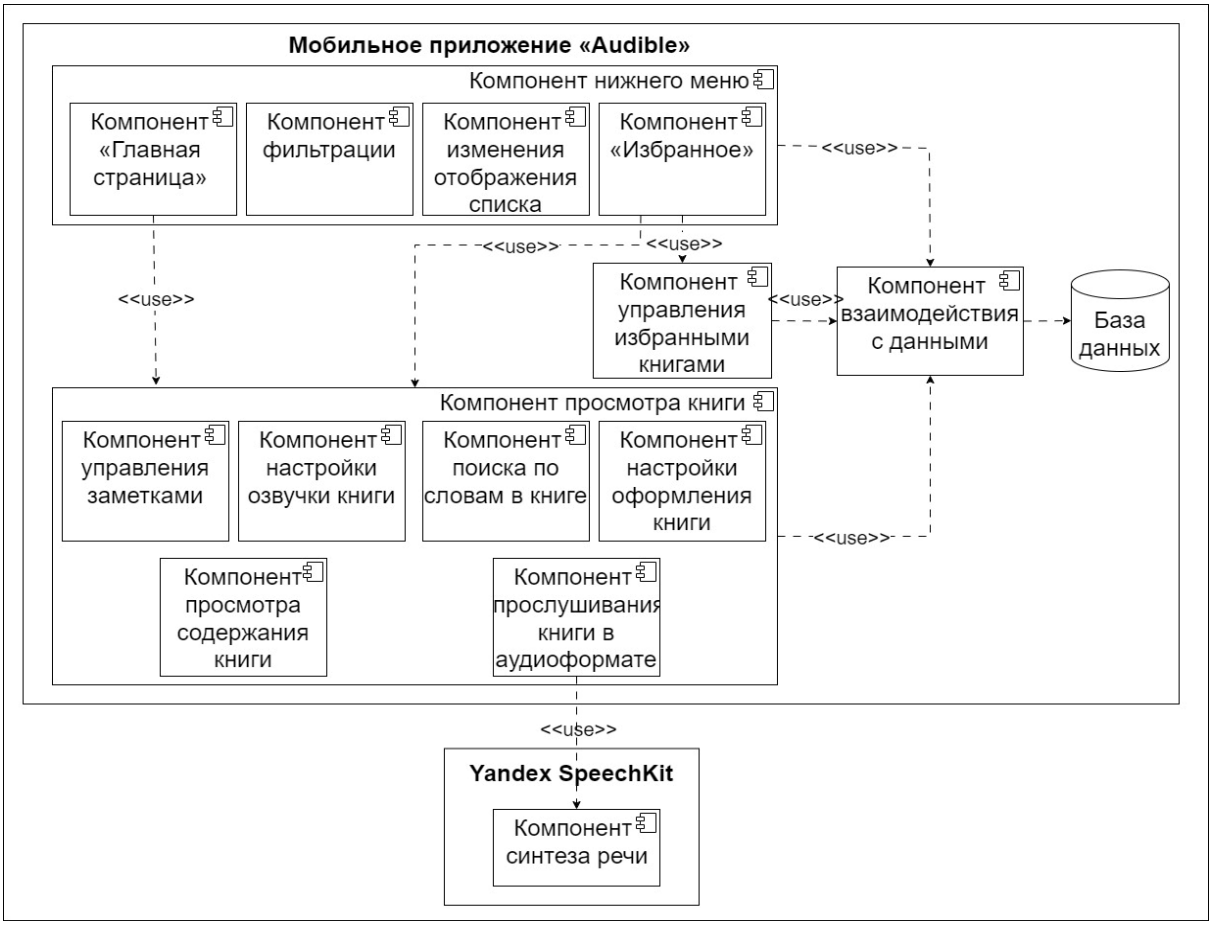


Рисунок 7 – Диаграмма компонентов приложения

Компонент нижнего меню позволяет переключаться между «Главной страницей» и страницей «Избранное». Также нижнее меню содержит кнопки для загрузки книг в приложение, фильтрации книг и изменения отображения списка книг.

Компонент «Главная страница» является точкой входа приложения и представляет из себя раздел, в котором пользователь может просматривать список загруженных книг. Компонент получает информацию из базы данных с помощью компонента взаимодействия с данными.

Компонент «Избранное» представляет из себя раздел приложения, в котором хранятся книги, выбранные пользователем для быстрого доступа к ним.

Компонент изменения отображения списка книг представляет из себя метод, который отвечает за визуальное изменение списка книг с плитки на структурированный список с полями названия книги и автора.

Компонент фильтрации представляет из себя метод, который отвечает за визуальное изменение списка книг: фильтрации по алфавиту.

Компонент управления избранными книгами представляет из себя методы для добавления книги в избранное и удаления ее из этого раздела. Используется компонентами «Главная страница» и «Избранное».

Компонент просмотра книги представляет из себя страницу приложения, в которой отображается текст книги, а также нижнее меню с прогрессом чтения, кнопками для перехода в аудиоформат и настройки оформления книги. Помимо этого, данная страница содержит кнопки для просмотра заметок, содержания и поиска по словам. Используется компонентами управления заметками, настройки озвучки книги, поиска по словам, настройки оформления, просмотра содержания, прослушивания книги в аудиоформате. Компонент получает информацию из базы данных с помощью компонента взаимодействия с данными.

Компонент управления заметками представляет из себя методы для добавления цитаты, просмотра списка цитат и удаления выбранной цитаты из раздела цитат. Данный компонент использует компонент просмотра книги.

Компонент настройки озвучки книги представляет из себя нижнее меню с двумя выпадающими списками, которые позволяют настроить скорость и голос озвучки. Использует компонент просмотра книги.

Компонент поиска по словам в книге представляет собой метод для ввода искомого слова и отображения результата поиска. Данный компонент использует компонент просмотра книги.

Компонент настройки оформления книги представляет из себя страницу приложения с диалоговым окном, которое содержит настройки темы

оформления книг, размера текста, размера отступов, включение автопрокрутки и выпадающий список для изменения скорости автопрокрутки. Данный компонент использует компонент просмотра книги.

Компонент просмотра содержания книги представляет из себя страницу приложения, в которой отображается содержание книги по главам. Данный компонент использует компонент просмотра книги.

Компонент прослушивания книги в аудиоформате представляет из себя метод, который позволяет прослушать книгу, озвученную с помощью нейросетевых технологий. Данный компонент использует компонент просмотра книги.

Компонент взаимодействия с данными представляет из себя класс, который отвечает за загрузку, хранение и использование данных. Также он содержит конфигурации для получения доступа к базе данных и интеграции базы данных.

2.4. Проектирование базы данных

На основе требований приложения для озвучки книг была спроектирована база данных. База данных состоит из 6 таблиц, в которых будет храниться информация о книгах, избранных книгах, заметках, прогрессе чтения, а также о настройках оформления книги и озвучки. На рисунке 8 представлена схема базы данных приложения.

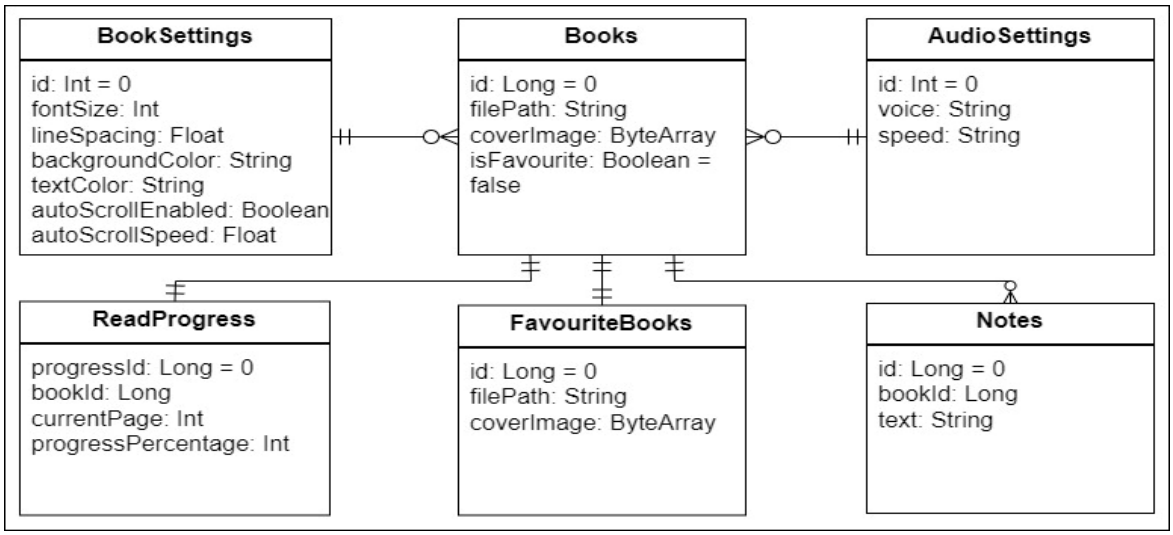


Рисунок 8 – Схема базы данных

Таблица Books содержит сведения о книгах, которые были загружены в приложение. Каждая запись в данной таблице включает уникальный идентификатор id, путь к файлу filePath, где хранится книга, изображение обложки книги coverImage в виде массива байтов, и флаг isFavourite, который указывает, является ли книга избранной.

Таблица FavouriteBooks хранит информацию о книгах, которые были добавлены пользователем в раздел избранного. Структура таблицы FavouriteBooks аналогична структуре Books, за исключением поля isFavourite.

Таблица ReadProgress содержит данные о прогрессе чтения для каждой книги. Данная таблица включает следующие поля: уникальный идентификатор progressId, внешний ключ bookId для хранения прогресса чтения конкретной книги, текущую страницу currentPage, на которой остановился пользователь, и прогресс чтения в процентах progressPercentage.

Таблица BookSettings хранит информацию о пользовательских настройках оформления книги. Таблица содержит в себе поля уникального идентификатора id, размера шрифта fontSize, размера отступов lineSpacing, цвета фона backgroundColor, цвета текста textColor, скорости прокрутки autoScrollSpeed, и флаг autoScrollEnabled, который определяет, была ли включена автопрокрутка страницы.

Таблица AudioSettings хранит информацию о пользовательских настройках озвучки. Таблица содержит в себе поля уникального идентификатора id, голоса озвучки voice и скорости озвучки speed.

Таблица Notes хранит информацию о сохраненных пользователем заметках. Таблица содержит в себе поля уникального идентификатора id, внешнего ключа bookId для хранения цитат конкретной книги.

2.5. Проектирование пользовательского интерфейса

Для создания интерфейса было выбрано несколько основных цветов, которые представлены на рисунке 9.

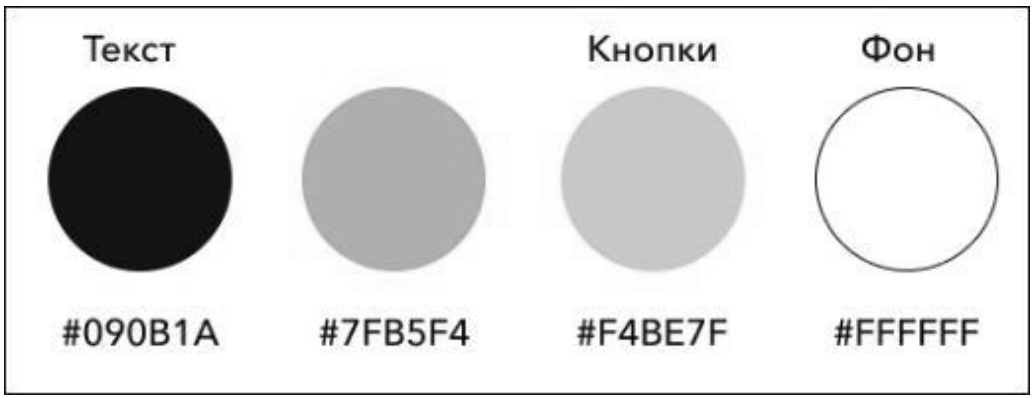


Рисунок 9 – Палитра основных цветов приложения «Audible»

Главный экран будет содержать список книг, загруженных в приложение, а также кнопки для открытия раздела избранных книг, фильтрации и изменения отображения списка. На рисунке 10 представлен макет главного экрана приложения.

Рисунок 10 – Макет главного экрана

При нажатии на кнопку изменения отображения внешний вид списка меняется с плитки на структурированный список, в котором также отображаются название книги и автор. При нажатии на кнопку фильтрации книги список книг изменяется в соответствии с выбранной фильтрацией по алфавиту.

При нажатии на кнопку с изображением сердца (раздел избранных книг) открывается экран со списком избранных книг, если книги были добавлены в избранное. Если список избранных книг пуст, то будет выведена надпись: «Здесь будут отображаться избранные книги».

На этом экране также доступны кнопки для фильтрации и изменения отображения списка книг.

Макеты экрана раздела избранных книг представлены на рисунке 11.

Рисунок 11 – Макет экрана раздела избранных книг

При нажатии на обложку книги в списке загруженных книг открывается экран с содержимым книги. Макет экрана детального просмотра книги представлен на рисунке 12.

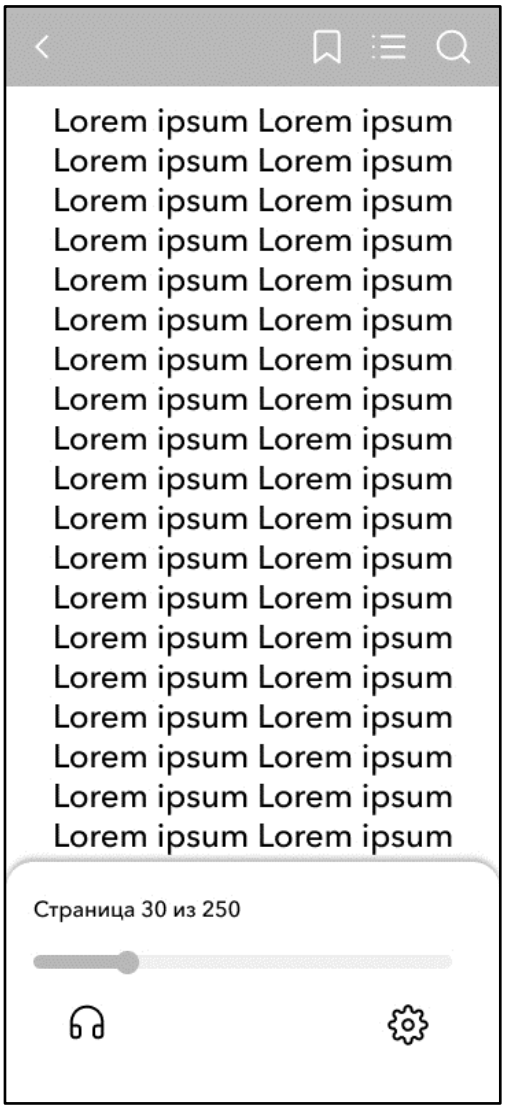
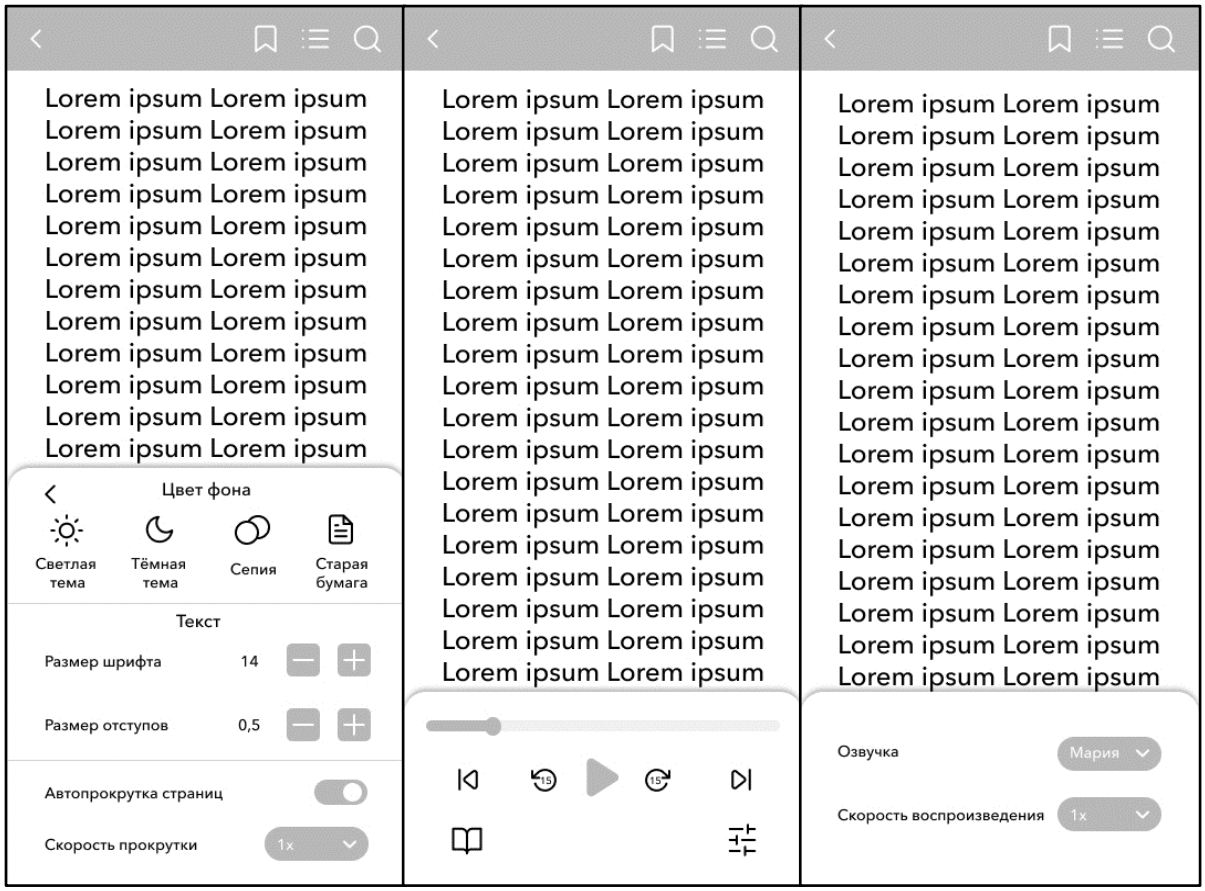


Рисунок 12 – Макет экрана детального просмотра книги

На экране детального просмотра книги содержатся кнопки для перехода на следующие экраны: заметки (сохраненные цитаты) и содержание книги. Помимо этого, на данном экране также расположена кнопка для поиска по словам в книге.

Экран детального просмотра содержит разные виды всплывающих меню: для настройки оформления книги, прослушивания книги в аудиоформате и настройки озвучки.

Макеты для всех вышеперечисленных видов нижнего меню отображены на рисунке 13.



а *б* *в*

Рисунок 13 – Макеты видов нижнего меню: *а* – для настройки оформления книги; *б* – для прослушивания книги; *в* – для настройки озвучки

Выводы по второй главе

В ходе проектирования были выявлены функциональные и нефункциональные требования к приложению, спроектирована диаграмма вариантов использования и описана спецификация вариантов использования. Также был выбран архитектурный подход для реализации мобильного приложения, спроектирована модель базы данных мобильного приложения и разработаны макеты пользовательского интерфейса.

3. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

3.1. Средства реализации

Приложение для озвучки книг было реализовано для операционной системы Android на языке Kotlin в среде разработки Android Studio. Для реализации пользовательского интерфейса был использован язык разметки XML [20].

XML – язык разметки, который используется для описания данных. Для создания какого-либо элемента в документе определяется тег, между парой тегов помещается необходимый элемент. В открывающем теге помещаются атрибуты (свойства) данного элемента. Для каждого типа элемента применяется своя пара тегов.

Для разработки реляционной локальной базы данных была использована СУБД SQLite. Для работы с SQLite была применена библиотека Room 2.6.1 [21]. Библиотека Room является высокоуровневым интерфейсом для SQLite и занимается созданием и поддержкой базы данных. Использование специальных аннотаций необходимо для корректной работы с SQLite.

Для реализации озвучки текста электронной книги использовался API Yandex SpeechKit [22]. В качестве параметров в теле запроса передается текст, который нужно озвучить, язык, голос озвучки, эмоция, скорость озвучки и формат аудио. Ответ представляет собой бинарное содержимое аудиофайла, формат которого зависит от значения параметра формата аудио в теле запроса.

Для передачи данных асинхронно использовались корутины [23]. Корутины (coroutines) – это блоки кода, которые выполняются асинхронно, не блокируя поток, из которого они запускаются. Корутины позволяют эффективно использовать ограниченное количество потоков для выполнения параллельных задач, сохраняя высокую производительность. Использование корутин необходимо для выполнения ресурсоемких операций в фоновом потоке во избежание блокировки основного потока, в котором выполняется управление пользовательским интерфейсом.

3.2. Архитектура

В соответствии с принятой архитектурой приложения были реализованы компоненты мобильного приложения.

Компонент View состоит из XML-файлов, описывающих визуальный интерфейс приложения. Данный компонент состоит из 13 файлов XML, где 5 файлов описывают интерфейс экранов приложения, 2 файла описывают интерфейс всплывающих окон, 4 файла отвечают за отображение элементов списка и 2 файла отвечают за меню. Все взаимодействия пользователя с интерфейсом обрабатываются с помощью компонента ViewModel.

Компонент ViewModel представляет собой набор классов, каждый из которых отвечает за обеспечение логики взаимодействия пользователя с определенным экраном приложения. Также этот компонент обеспечивает поддержку жизненного цикла соответствующего экрана. Для компонента ViewModel было реализовано 4 класса.

Компонент Model содержит в себе логику работы с данными, а также модели данных, которые необходимы для работы приложения. Данный компонент состоит из 6 классов: 1 класс служит для структурирования информации, а другие 5 классов – для взаимодействия с локальной базой данных. Всего в системе взаимодействуют 13 классов. На рисунке 14 изображена диаграмма классов приложения для озвучки книг.

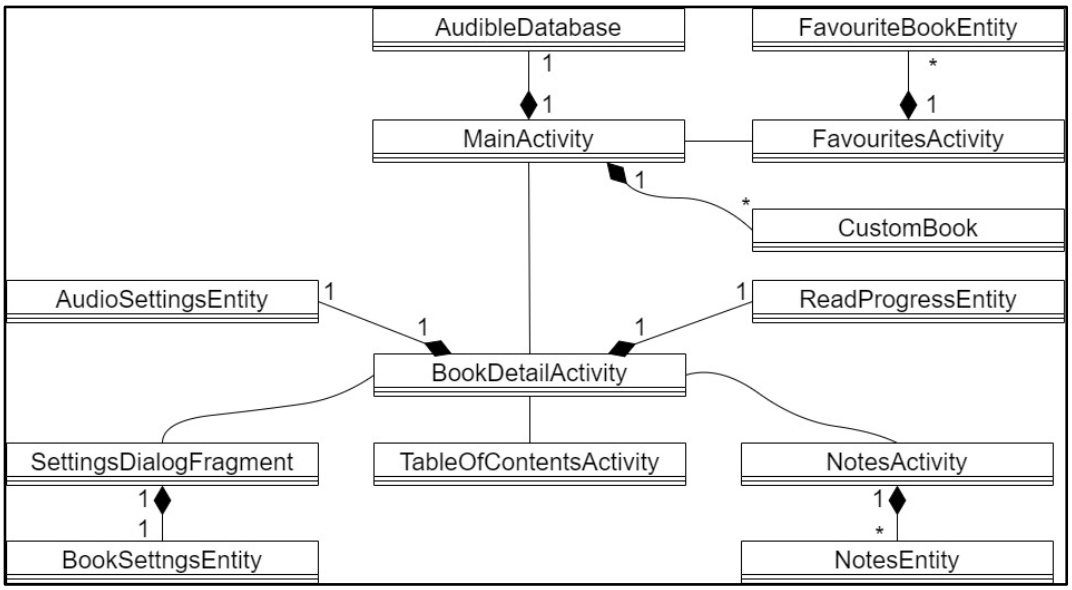


Рисунок 14 – Диаграмма классов приложения

Главным классом приложения является класс `MainActivity`, с которого происходит открытие приложения. Если пользователь уже загружал книги в приложение, то происходит загрузка главного экрана и данных о загруженных книгах из локальной базы данных. В `MainActivity` также реализованы возможности вызова методов для загрузки книги в приложение, удаления книги из приложения, добавления книги в избранное, фильтрации книг и изменения отображения списка книг.

Класс `AudibleDatabase` отвечает за взаимодействие с базой данных. Данный класс является абстрактным классом и определяет абстрактные методы, которые возвращают DAO для взаимодействия с соответствующими таблицами базы данных. В свою очередь DAO содержат методы для выполнения операций с базой данных.

Класс `CustomBook` служит для структурирования информации о книге.

Класс `FavouriteBookEntity` служит для структурирования информации о избранных книгах.

Класс `ReadProgressEntity` служит для структурирования информации о прогрессе чтения конкретной книги.

Класс `NotesEntity` служит для структурирования информации о сохраненных цитатах конкретной книги.

Класс `BookSettingsEntity` служит для структурирования информации о настройках оформления книг.

Класс `AudioSettingsEntity` служит для структурирования информации о настройках озвучки книг.

Класс `FavouritesActivity` отвечает за загрузку экрана с избранными книгами. В `FavouritesActivity` реализована возможность вызова метода для удаления книги из избранного.

Класс `BookDetailActivity` отвечает за загрузку экрана просмотра книги. В `BookDetailActivity` реализованы возможности вызова методов

для прослушивания книги в аудиоформате, изменения настроек озвучки, сохранения цитат, сохранения прогресса чтения и поиска по словам.

Класс `SettingsDialogFragment` отвечает за загрузку диалогового окна настроек оформления книги. В `SettingsDialogFragment` реализованы возможности вызова методов для изменения темы книги, размера шрифта, размера отступов, включения и выключения автопрокрутки, изменения скорости автопрокрутки.

Класс `TableOfContentsActivity` отвечает за загрузку экрана содержимого книги.

Класс `NotesActivity` отвечает за загрузку экрана с сохраненными цитатами в книге. В `NotesActivity` реализована возможность вызова метода для удаления цитаты.

3.3. Реализация локальной базы данных

В соответствии с разработанной схемой базы данных были реализованы классы, которые являются моделями данных, хранящимися в соответствующих таблицах базы данных. Данные классы помечаются аннотацией `@Entity` для переопределения свойства создания таблицы. В листинге 1 приведен пример создания таблицы для хранения цитат.

Листинг 1 – Реализация создания таблицы для хранения цитат

```
@Entity(tableName = "notes", foreignKeys = [ForeignKey(entity =
BookEntity::class, parentColumns = ["id"], childColumns = ["bookId"],
onDelete = ForeignKey.CASCADE)])

data class NotesEntity(
    @PrimaryKey(autoGenerate = true)
    val id: Long = 0,
    val bookId: Long,
    val text: String
)
```

В данном классе используется внешний ключ `bookId` для привязки сохраненных цитат к конкретной книге. Для реализации первичного ключа используется аннотация `@PrimaryKey` с указанием в качестве параметра автоматической генерации первичного ключа.

Для работы с таблицами базы данных также были созданы интерфейсы, которые помечаются аннотацией `@Dao`. Данные интерфейсы содержат в себе методы для работы с базой данных. В листинге 2 представлена реализация интерфейса для работы с таблицей заметок.

Листинг 2 – Реализация интерфейса для работы с таблицей заметок

```
@Dao
interface NotesDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertNote(note: NotesEntity)

    @Query("SELECT * FROM notes WHERE bookId = :bookId")
    suspend fun getNoteByBookId(bookId: Long): List<NotesEntity>

    @Query("SELECT * FROM notes WHERE bookId = :bookId AND text = :text")
    suspend fun getNoteByTitle(bookId: Long, text: String): NotesEntity?

    @Query("DELETE FROM notes WHERE id = :id")
    suspend fun deleteNote(id: Long)}
```

В реализации интерфейса для работы с таблицей заметок используется аннотация `@Insert` для вставки данных, при этом также применяется стратегия `OnConflictStrategy.REPLACE` для замены существующего объекта на новый при конфликте. Аннотация `@Query` необходима для выполнения SQL-запросов. В параметрах данной аннотации указывается сам SQL-запрос, например, для получения всех заметок текущей книги.

Для описания базы данных был создан абстрактный класс `AudibleDatabase` с аннотацией `@Database`, в параметрах которой указаны все сущности базы данных, а также ее версия. В листинге 3 приведена реализация класса базы данных с примером применения миграций.

Листинг 3 – Реализация класса `AudibleDatabase`

```
@Database(entities = [BookEntity::class, FavouriteBookEntity::class, Read-
ProgressEntity::class, BookSettingsEntity::class, AudioSettings-
Entity::class, NotesEntity::class],version = 15, exportSchema = false)

abstract class AudibleDatabase : RoomDatabase() {
    abstract fun bookDao(): BookDao
    abstract fun favouriteBookDao(): FavouriteBookDao
    abstract fun readProgressDao(): ReadProgressDao
    abstract fun bookSettingsDao(): BookSettingsDao
    abstract fun audioSettingsDao(): AudioSettingsDao
    abstract fun notesDao(): NotesDao

    private fun createDatabase(context: Context): AudibleDatabase {
        return Room.databaseBuilder(
```

```
            context.applicationContext,
            AudibleDatabase::class.java,
            "audible-database"
        ).addMigrations(MIGRATION_1_2).build()
    }

    private val MIGRATION_1_2: Migration = object : Migration(1, 2){
        override fun migrate(database: SupportSQLiteDatabase){
            database.execSQL(
                "CREATE TABLE IF NOT EXISTS `favouriteBooks`(
                    `id` INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
                    `filePath` TEXT NOT NULL,
                    `coverImage` BLOB NOT NULL)")
        }
    }
}
```

3.4. Реализация загрузки книги в приложение

На главном экране приложения реализована возможность для загрузки книг в приложение с устройства. Алгоритм загрузки книг представлен на рисунке 15.

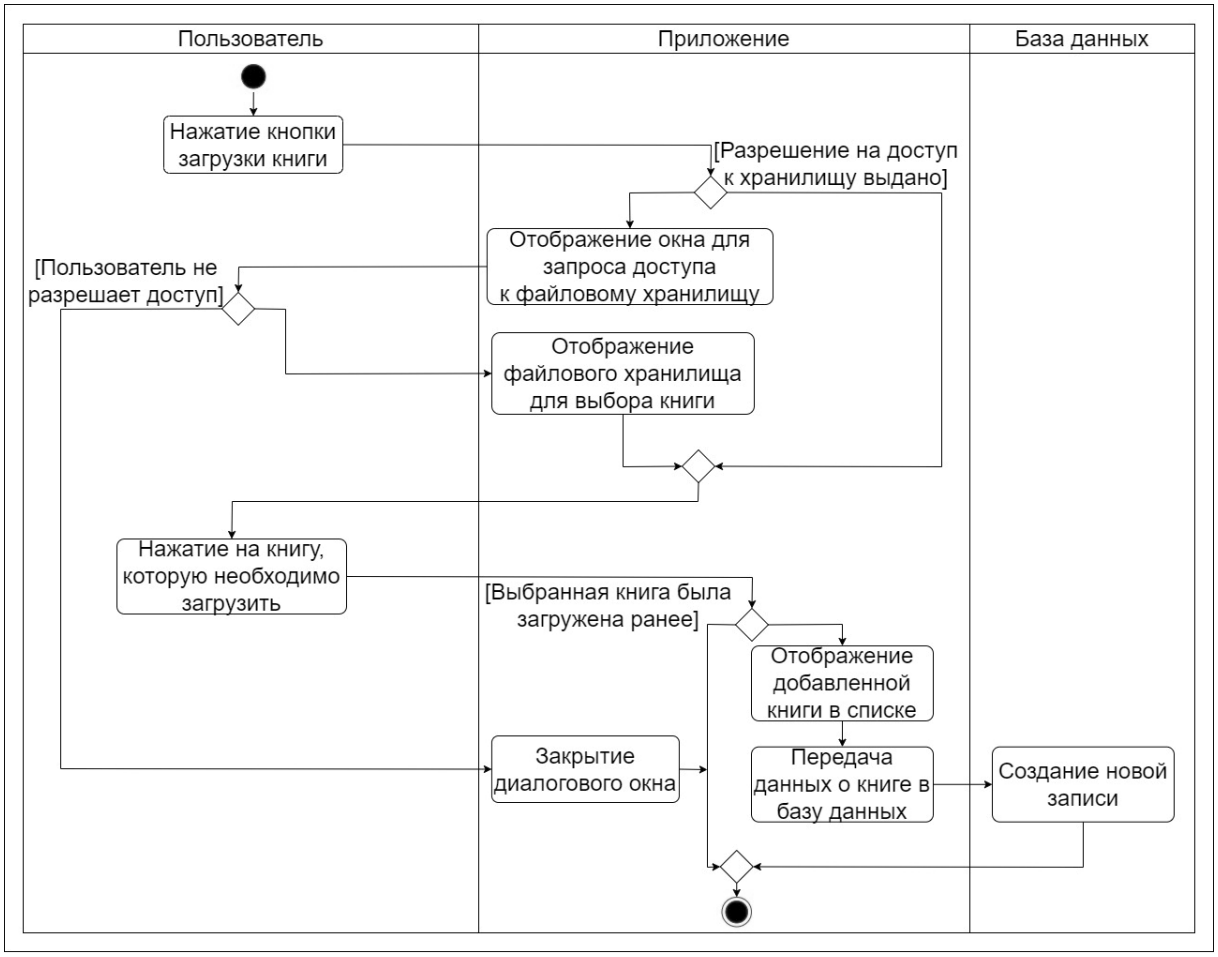


Рисунок 15 – Диаграмма деятельности алгоритма загрузки книг

Механизм загрузки книги в приложение начинается, когда пользователь нажимает кнопку для загрузки книги. Если доступ к файловому хранилищу был выдан ранее, то приложение отображает пользователю файловое хранилище для выбора книг. Если же доступ не был выдан ранее, то приложение отображает диалоговое окно для запроса доступа к файловому хранилищу. Пользователь может разрешить и запретить доступ. Если пользователь запретил доступ к файловому хранилищу, то диалоговое окно закрывается и действие заканчивается, иначе открывается файловое хранилище. После открытия файлового хранилища и нажатия пользователем на нужную книгу приложение проверяет, была ли эта книга загружена ранее. Если выбранная пользователем книга была загружена ранее, то действие заканчивается. Если же выбранная книга не была загружена ранее, то приложение отображает эту книгу в списке книг, а также передает данные о книге в базу данных. После этого в базе данных создается новая запись о добавленной книге, затем действие заканчивается.

Функция, реализующая добавление книги в список, представлена в листинге 4.

Листинг 4 – Функция добавления книги в список

```
private fun addBookToBookList(newBook: CustomBook, bookList: Mutable-
List<CustomBook>) {
    val bookExists = bookList.any { it.filePath == newBook.filePath }

    if (!bookExists) {
        bookViewModel.refreshBooks()
        bookAdapter.notifyDataSetChanged()
    } else
        Toast.makeText(this, "Эта книга уже была загружена",
        Toast.LENGTH_LONG).show()
}
```

3.5. Реализация просмотра списка книг

При запуске мобильного приложения на главном экране отображается список раннее загруженных книг. При открытии главного экрана `MainActivity` создается экземпляр класса `BookViewModel`, который вызывает метод `getBooks` из экземпляра класса `BookRepositoryImpl` через интерфейс

BookRepository. Метод getBooks извлекает из локальной базы данных информацию о загруженных книгах, обрабатывает эту информацию и возвращает ее в BookViewModel в виде LiveData. Затем BookViewModel использует MutableLiveData для передачи обработанных данных для отображения на экране MainActivity.

В листинге 5 представлена часть кода класса MainActivity, которая получает данные от BookViewModel с помощью MutableLiveData.

Листинг 5 – Реализация части кода главного экрана

```
bookViewModel.bookData.observe(this as LifecycleOwner) { books ->
    bookList.clear()
    bookList.addAll(books)
    bookAdapter.notifyDataSetChanged()}
```

В данном коде показан экземпляр класса BookViewModel, который следит за обновлениями данных bookData, которые являются LiveData в BookViewModel. Если данные изменились, то очищаем список и добавляем в него актуальные данные, а также уведомляем адаптер об изменениях для обновления отображения списка на экране MainActivity.

В листинге 6 представлена часть кода класса BookViewModel с использованием MutableLiveData для обработки данных, которые были получены из BookRepositoryImpl через интерфейс BookRepository.

Листинг 6 – Реализация части кода класса BookViewModel

```
class BookViewModel(application: Application,  private val bookRepository:
BookRepository) : ViewModel() {

    private val _bookData = MutableLiveData<List<CustomBook>>()
    val bookData: LiveData<List<CustomBook>> get() = _bookData

    fun refreshBooks() {
        bookRepository.getBooks().observeForever { books ->
            _bookData.postValue(books)
        }
    }
}
```

В приведенном коде описан метод для обновления данных о книгах. В методе refreshBooks используется экземпляр класса BookRepository для получения списка книг. В данном методе мы наблюдаем за LiveData, и

при изменении данных передаем их в _bookData для уведомления наблюдателей. В переменную bookData присваивается значение _bookData.

В листинге 7 отображен метод getBooks, который реализован в классе BookRepositoryImpl. Данный метод отвечает за получение данных о загруженных в приложение книгах из локальной базы данных.

Листинг 7 – Реализация метода getBooks в BookRepositoryImpl

```
override fun getBooks(): LiveData<List<CustomBook>> {
    val liveData = MutableLiveData<List<CustomBook>>()
    CoroutineScope(Dispatchers.IO).launch {
        val bookEntities = bookDao.getAllBooks()
        val customBooks = bookEntities.mapNotNull { bookEntity ->
            val book = parseEpubFile(bookEntity.filePath)
            if (book != null) {
                val progress = readProgressDao
                .getProgressByBookId(bookEntity.id)?
                .progressPercentage ?: 0
                CustomBook(book, bookEntity.filePath,
                bookEntity.isFavourite, progress)
            } else {
                null
            }
        }
        withContext(Dispatchers.Main) {
            liveData.postValue(customBooks)
        }
    }
    return liveData
}
```

Приведенный метод возвращает список книг класса CustomBook в виде LiveData. В методе getBooks используется CoroutineScope (область применения корутин), так как все корутины должны работать в определенной области для обеспечения возможности управлять ими как группами. Это позволяет избежать утечек: корутины не будут продолжать работать в фоновом режиме, когда они больше не нужны. CoroutineScope объявляет диспетчера (Dispatcher), который используется для запуска корутин.

В методе getBooks используется Dispatchers.IO, предназначенный для корутин, выполняющих операции с базой данных. После определения области применения корутин вызывается метод getAllBooks из DAO-интерфейса bookDao для получения информации о всех записях о книгах в базе данных. Далее из каждой книги извлекаются необходимые данные.

Прогресс чтения получаем с помощью метода `getProgressByBookId` из интерфейса `readProgressDao`, а затем создаем экземпляр класса `CustomBook`, заполняем его извлеченными данными для каждой книги. С помощью конструктора `withContext`, который позволяет запускать корутину в контексте, отличном от контекста родительской корутины, обновляем данные `LiveData` в главном потоке, так как использование главного потока необходимо для обновления пользовательского интерфейса.

Для реализации отображения списка книг используется элемент `RecyclerView`. Каждый элемент списка представляет собой обложку книги, прогресс чтения, кнопку для добавления в избранное и удаления книги из приложения. Для того, чтобы реализовать такое отображение у каждого элемента `RecyclerView` был создан отдельный файл разметки item_book.xml.

Для заполнения списка элементами используется класс `BookAdapter`. Этот класс связывает данные о книгах с элементами списка `RecyclerView`. При добавлении или удалении книг из приложения список корректно обновляется с учетом измененных данных, а также происходит обновление прогресса чтения и отображения, добавлена ли книга в избранное.

Часть реализации класса `BookAdapter` приведена в листинге 8.

Листинг 8 – Реализация части класса `BookAdapter`

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ViewHolder {
    val view = LayoutInflater.from(parent.context).inflate(itemBook, par-
ent, false)
    return ViewHolder(view)
}

override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    val customBook = bookDataClassList[position]
    holder.cover.setImageBitmap(getCoverImage(customBook.epubBook))
    holder.cover.setOnClickListener {
        openBook(customBook.filePath, it)
    }
    bookViewModel?.loadReadingProgress(customBook.filePath)?.observe(
    holder.itemView.context as LifecycleOwner){
        progress -> holder.readingSeekBar.progress = progress ?: 0
        holder.readingSeekBar.isEnabled = false
        holder.progressPercentage.text = "${progress ?: 0}%"
    }
}
```

В приведенном коде метод `onCreateViewHolder` отвечает за создание объектов `ViewHolder` с макетом item_book.xml, когда элемент `RecyclerView` нуждается в этом. Компонент `ViewHolder` используется в `RecyclerView` для оптимизации производительности и плавности прокрутки списка.

Метод `onBindViewHolder` отвечает за привязку данных к указанному `ViewHolder`. Этот метод вызывается для каждого элемента списка, когда элемент становится видимым на экране. В данном методе для каждого элемента списка устанавливается обложка и прогресс чтения конкретной книги. В листинге 9 приведен пример использования экземпляра класса `BookAdapter`.

Листинг 9 – Пример использования экземпляра класса `BookAdapter`

```
bookAdapter = BookAdapter(ContextType.MAIN_ACTIVITY, bookList, this,
bookViewModel, null)
binding.recyclerViewEpubFiles.layoutManager = GridLayoutManager(this, 3)
binding.recyclerViewEpubFiles.adapter = bookAdapter

bookViewModel.bookData.observe(this as LifecycleOwner) { books ->
    bookList.clear()
    bookList.addAll(books)
    bookAdapter.notifyDataSetChanged()
}
```

В данном коде определяется отображение элементов списка в виде плитки, а также устанавливается адаптер для `RecyclerView`. Далее отслеживаются изменения данных `BookViewModel` и уведомляется адаптер о необходимости обновления списка.

3.6. Реализация озвучки книги

Для озвучки с помощью нейросетевых технологий было использовано Yandex SpeechKit API. В листинге 10 приведена реализация метода `synthesizeSpeechAndPlay`.

Листинг 10 – Реализация метода для озвучки книги

```
private suspend fun synthesizeSpeechAndPlay(text: String, voice: String,
speed: String) {
    if (text.isNotBlank()) {
        val synthesisRequestBody = mapOf(
            "text" to text,
            "voice" to voice,
```

```
            "format" to "mp3",
            "folderId" to "b1g1ddlkc9drq0ca0cf2",
            "lang" to "ru-RU",
            "speed" to speed,
            "emotion" to "neutral"
        )
        val encodedBody = synthesisRequestBody.entries.joinToString("&"){
            "${URLEncoder.encode(it.key, "UTF-8")}=${URLEncoder.en-
code(it.value, "UTF-8")}"
        }
        val contentType = "application/x-www-form-urlencoded".toMediaType()
        val requestBody = encodedBody.toRequestBody(contentType)
        val client = OkHttpClient()
        val request = Request.Builder()
            .url("https://tts.api.cloud.yandex.net/speech/v1/tts:synthe-
size")
            .addHeader("Authorization", "Bearer $IAM_TOKEN")
            .post(requestBody)
            .build()
        try {
            val response = withContext(Dispatchers.IO) { client.newCall(re-
quest).execute() }
            val audioContent = response.body?.bytes() ?: throw IOExcep-
tion("No response body")
            val outputFile = File(filesDir, "output.mp3")
            FileOutputStream(outputFile).use {
                it.write(audioContent)
            }
            withContext(Dispatchers.Main) {
                mediaPlayer.reset()
                mediaPlayer.setDataSource(outputFile.absolutePath)
                mediaPlayer.prepare()
                mediaPlayer.start()
            }
        } catch (e: Exception) {
            Log.e("SpeechKit", "Synthesis error", e)
        }
    }
}
```

В данный метод передаются текст для озвучки, голос озвучки и скорость озвучки. Эти параметры передаются в тело запроса. Помимо этого, в теле запроса указывается формат, в который преобразуется аудио, и язык озвучки. В заголовке запроса указывается заранее сгенерированный токен. Далее полученный файл передается в `MediaPlayer` для воспроизведения аудиофайла.

Для извлечения текста текущей страницы из `WebView` используется метод `onTextExtracted`. Данный метод является переопределенным методом интерфейса `WebViewListener`, который реализуется в классе `BookDetailActivity`. Далее необходимо добавить `JavascriptInterface` для `WebView`, где будет вызван метод `onTextExtracted`. Для корректной работы

нужно также создать WebViewClient и в методе onPageFinished выполнить Javascript-код. Частичная реализация данного кода представлена в листинге 11.

Листинг 11 – Реализация части кода для извлечения текста из WebView

```
webView.addJavascriptInterface(object {
    @JavascriptInterface
    fun onTextExtracted(text: String) {
        webViewListener.onTextExtracted(text)
        webViewListener.onWebViewCreated(webView)
    }
}, "AndroidInterface")

webView.webViewClient = object : WebViewClient() {
    override fun onPageFinished(view: WebView, url: String) {
        view.postDelayed({
            view.loadUrl("javascript:window.AndroidInterface.onTextEx-
tracted(document.body.innerText);")
        }, 500)
    }
}
```

3.7. Реализация пользовательского интерфейса

В соответствии с разработанными макетами был реализован интерфейс приложения. Для всех классов, отвечающих за загрузку экранов, были созданы XML-файлы, описывающие визуальные части пользовательского интерфейса. Также были созданы XML-файлы, которые описывают разметку диалогового окна для настройки оформления книги, всплывающих окон, отображения элементов списка и меню.

На главном экране приложения расположен список загруженных книг, а также нижнее меню с кнопками для загрузки книги в приложение, перехода на экран избранных книг, изменения отображения списка с плитки на структурированный список и фильтрации. Для реализации списка был использован элемент RecyclerView, для кнопок – ImageButton. Верстка главного экрана приложения представлена в листинге 1 приложения.

Также была создана верстка для каждого элемента списка RecyclerView. Каждый элемент представляет собой обложку (элемент ImageView), кнопки ImageButton для добавления в избранное, удаления книги из приложения и элемент SeekBar для отображения прогресса чтения. Помимо

этого, в данном файле верстки также содержатся элементы `TextView` для отображения автора и названия книги в случае, если книга отображается в виде структурированного списка. Верстка элемента списка `RecyclerView` представлена в листинге 2 приложения.

Скриншоты главной страницы представлены на рисунке 16.

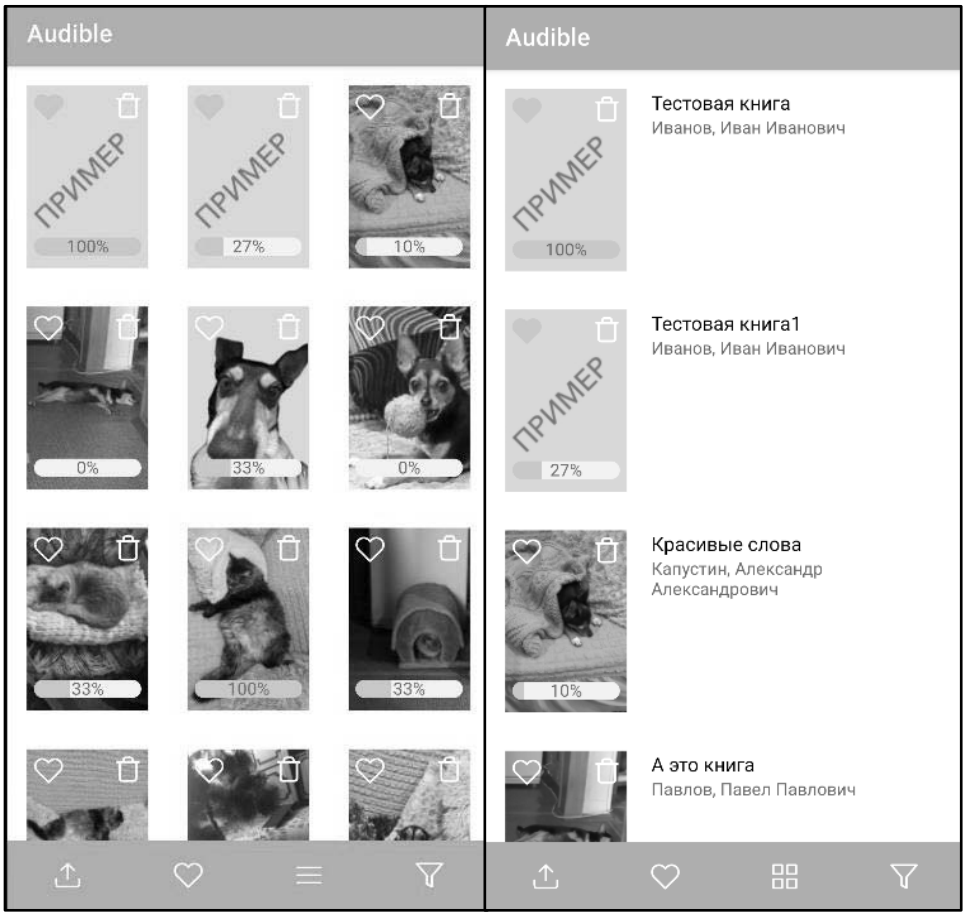


Рисунок 16 – Скриншот главной страницы приложения

Страница для чтения книги состоит из элементов `WebView` для отображения текста и `ViewPager` для постраничного перелистывания книги. Также для реализации всплывающего меню был использован элемент `LineaerLayout` с указанием в атрибуте `layout_behaviour` поведения всплывающего нижнего меню.

Данное меню содержит в себе элементы `TextView` для отображения текущей страницы и общего количества страниц, `SeekBar` для отображения прогресса чтения и возможности перелистывания страниц с помощью этого

элемента, а также `ImageButton` кнопки для перехода к прослушиванию книги в аудиоформате и открытия диалогового окна для настройки оформления книги. Разметка страницы для просмотра книги представлена в листинге 3 приложения.

При нажатии на кнопку для настройки оформления книги открывается диалоговое окно, которое содержит `ImageButton` кнопки для выбора темы, изменения размера шрифта и отступов.

В диалоговом окне возле каждого элемента отображается текст, который реализуется с помощью элемента `TextView`.

Также имеется элемент `Spinner`, который является выпадающим списком, и используется для выбора скорости прокрутки. Элемент `SwitchCompat`, является кнопкой-переключателем для включения или выключения автопрокрутки. Скриншот страницы для просмотра книги представлен на рисунке 17.

Рисунок 17 – Скриншот страницы просмотра книги

Скриншот разметки диалогового окна для настройки оформления книги представлен на рисунке 18.

Рисунок 18 – Скриншот диалогового окна для настройки оформления

При нажатии на кнопку для перехода к прослушиванию книги в аудиоформате всплывающее нижнее меню меняется. Данное меню содержит в себе кнопки для паузы, воспроизведения, перемотки к следующей/предыдущей странице, перемотки вперед/назад на 15 секунд, `SeekBar` для отображения прогресса чтения, `ImageButton` кнопки для возврата к режиму чтения и настроек озвучки. Скриншот разметки нижнего меню для управления прослушиванием книги представлен на рисунке 19.

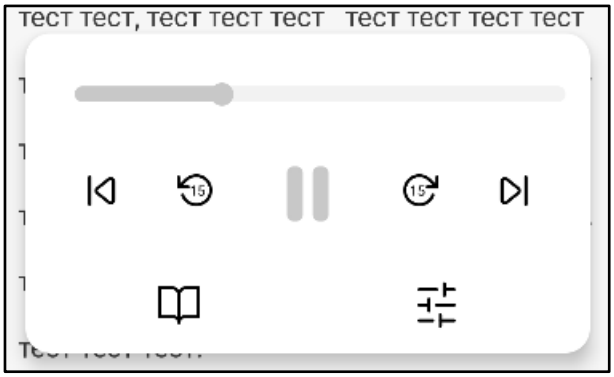


Рисунок 19 – Скриншот нижнего меню для управления прослушиванием книги

На странице просмотра книги при выделении текста появляется всплывающее меню, которое содержит пункт «Сохранить цитату». В листинге 12 представлена разметка всплывающего меню для сохранения цитаты.

Листинг 12 – Разметка всплывающего меню для сохранения цитаты

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:id="@+id/save_to_notes"
        android:title="Сохранить цитату"
        app:showAsAction="always" />
</menu>
```

Скриншот всплывающего меню для сохранения цитаты представлен на рисунке 20.

Рисунок 20 – Скриншот всплывающего меню для сохранения цитаты

Сохраненные цитаты расположены на странице раздела «Заметки». Для отображения цитат в виде списка используется элемент `RecyclerView`, каждый элемент которого представляет из себя `TextView` и кнопки `ImageButton` для удаления цитаты. Скриншот страницы раздела «Заметки» представлен на рисунке 21.

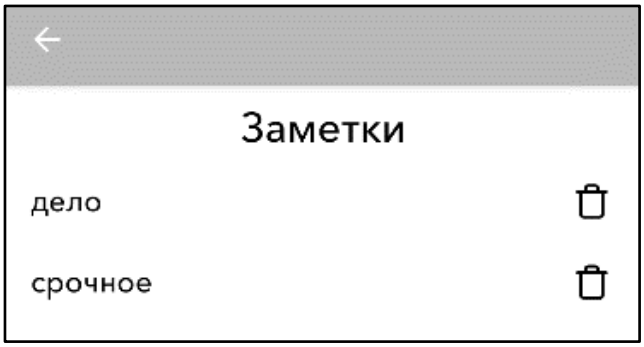


Рисунок 21 – Скриншот страницы раздела «Заметки»

Страница раздела «Содержание» имеет аналогичную разделу «Заметки» верстку за исключением кнопки удаления. Каждый элемент списка `RecyclerView` представляет собой текст, который реализован с помощью `TextView`. Скриншот страницы раздела «Содержание» представлен на рисунке 22.

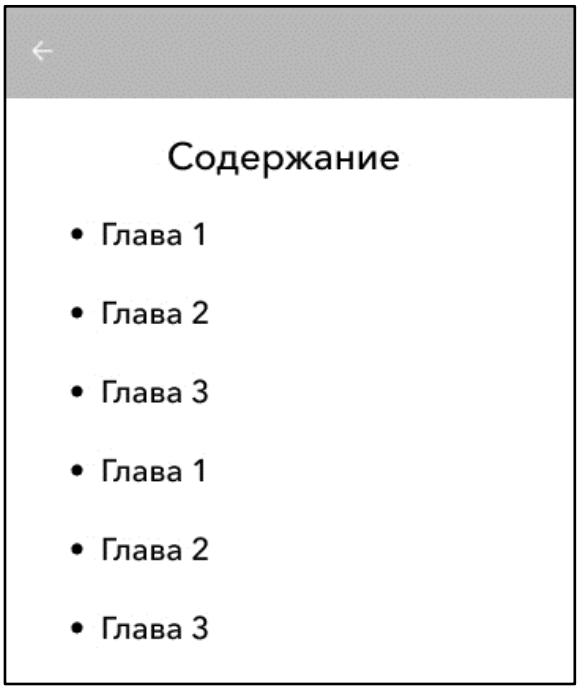


Рисунок 22 – Скриншот страницы раздела «Содержание»

Вывод по третьей главе

В данной главе была описана реализация приложения: средства реализации, реализация архитектуры мобильного приложения, реализация локальной базы данных, а также реализация компонентов и пользовательского интерфейса.

4. ТЕСТИРОВАНИЕ

В рамках тестирования разрабатываемого Android-приложения использовалось устройство Pixel 5.

Для тестирования мобильного приложения было выбрано функциональное тестирование [24]. Функциональное тестирование проверяет работоспособность функциональных возможностей приложения, которые были представлены в качестве списка функциональных требований при проектировании системы. В таблице 1 представлен набор тестов основных аспектов работы приложения.

Таблица 1 – Набор тестов для функционального тестирования

№	Название теста	Действие	Ожидаемый результат	Тест пройден?
1	Проверка работы функции загрузки книги	Запустить приложение, нажать кнопку загрузки книги, дать разрешение на доступ к файловому хранилищу, нажать на нужную книгу	Загруженная книга отображается в списке книг	Да
2	Проверка работы функции удаления книги	Запустить приложение с заранее загруженными книгами, нажать кнопку удаления книги	Выбранная книга удаляется из списка, список обновляется и отображает все книги, кроме удаленной	Да
3	Проверка работы функции фильтрации по алфавиту	Запустить приложение с заранее загруженными книгами, нажать кнопку фильтрации	Книги изменяют положение в списке в соответствии с сортировкой (от А до Я или от Я до А), изменяется внешний вид кнопки фильтрации для отображения текущего способа фильтрации	Да
4	Проверка работы функции добавления книг в раздел избранного	Запустить приложение с заранее загруженными книгами, нажать кнопку сердца для добавления в избранное	Кнопка сердца на выбранной книге изменяет цвет для отображения того, что книга находится в избранном, книга добавляется в избранное и отображается на экране избранных книг	Да

Продолжение таблицы 1

№	Название теста	Действие	Ожидаемый результат	Тест пройден?
5	Проверка работы функции просмотра списка избранных книг	Запустить приложение с заранее загруженными книгами, нажать кнопку для добавления книги в избранное, нажать кнопку для открытия экрана избранных книг	Добавленные в избранное книги отображаются в списке избранных книг	Да
6	Проверка работы функции удаления книг из раздела избранного	Запустить приложение с заранее загруженными книгами, нажать кнопку для добавления книги в избранное, нажать кнопку для открытия экрана избранных книг, нажать кнопку удаления из избранного	Выбранная книга удаляется из списка на экране избранных книг, на главном экране кнопка сердца для добавления в избранное изменяет цвет и становится снова активной	Да
7	Проверка работы функции просмотра списка книг	Запустить приложение с заранее загруженными книгами	Отображается список загруженных книг	Да
8	Проверка работы функции изменения отображения списка книг	Запустить приложение с заранее загруженными книгами, нажать кнопку для изменения отображения списка книг	Список загруженных книг меняется с плитки на структурированный список, при повторном нажатии меняется снова на плитку	Да
9	Проверка работы функции просмотра содержимого книги	Запустить приложение с заранее загруженными книгами, нажать на обложку книги	Открывается экран с текстом книги	Да
10	Проверка работы функции сохранения прогресса прочтения книги	Запустить приложение с заранее загруженными книгами, нажать на обложку книги, пролистать несколько страниц, вернуться на главный экран	Прогресс чтения текущей книги сохраняется в базе данных, список книг обновляется и отображает актуальный прогресс прочтения книги	Да

Продолжение таблицы 1

№	Название теста	Действие	Ожидаемый результат	Тест пройден?
11	Проверка работы функции сохранения цитат	Запустить приложение с заранее загруженными книгами, нажать на обложку книги, выделить необходимый текст, во всплывшем меню нажать на «Сохранить цитату»	Выбранный текст сохраняется в базу данных и отображается на экране с цитатами	Да
12	Проверка работы функции просмотра списка сохраненных цитат	Запустить приложение с заранее загруженными книгами и сохраненными цитатами, нажать на обложку книги, нажать на кнопку открытия экрана с цитатами	Отображается список с сохраненными цитатами в текущей книге	Да
13	Проверка работы функции удаления цитат	Запустить приложение с заранее загруженными книгами и сохраненными цитатами, нажать на обложку книги, нажать на кнопку открытия экрана с цитатами, нажать на кнопку удаления цитаты	Выбранная цитата удаляется из списка, отображается обновленный список с цитатами без удаленной цитаты	Да
14	Проверка работы функции настройки оформления книг	Запустить приложение с заранее загруженными книгами, нажать на обложку книги, нажать кнопку настроек, изменить какую-либо часть настроек	В зависимости от измененной настройки отображаются изменения, эти изменения сохраняются и применяются ко всем книгам	Да
15	Проверка работы функции поиска по словам в книге	Запустить приложение с заранее загруженными книгами, нажать на обложку книги, нажать кнопку поиска, ввести искомое слово	Совпадающие с искомым слова выделяются цветом, происходит переход к совпадению	Да

Окончание таблицы 1

№	Название теста	Действие	Ожидаемый результат	Тест пройден?
16	Проверка работы функции прослушивания книги в аудиоформате	Запустить приложение с заранее загруженными книгами, нажать на обложку книги, нажать кнопку наушников для перехода в режим озвучки	Отправляется запрос на озвучку, после получения ответа воспроизводится сгенерированный аудио файл	Да
17	Проверка работы функции настройки скорости озвучивания книги	Запустить приложение с заранее загруженными книгами, нажать на обложку книги, нажать кнопку наушников для перехода в режим озвучки, нажать кнопку настроек, выбрать из списка скорость озвучки книги	Отправляется запрос на озвучку с измененным параметром скорости озвучки, после получения ответа воспроизводится сгенерированный аудио файл	Да
18	Проверка работы функции настройки голоса озвучки книги	Запустить приложение с заранее загруженными книгами, нажать на обложку книги, нажать кнопку наушников для перехода в режим озвучки, нажать кнопку настроек, выбрать из списка голос озвучки книги	Отправляется запрос на озвучку с измененным параметром голоса озвучки, после получения ответа воспроизводится сгенерированный аудио файл	Да

Вывод по четвертой главе

В четвертой главе было проведено функциональное тестирование. Все тесты из набора были успешно пройдены. Можно сделать вывод, что разработанное приложение соответствует требованиям.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было реализовано Android-приложение для озвучивания книг.

В ходе выполнения выпускной квалификационной работы бакалавра были решены следующие задачи.

1. Произведен обзор предметной области и существующих решений.
2. Спроектирована архитектура мобильного приложения.
3. Реализовано программное обеспечение для озвучивания книг.
4. Проведено тестирование мобильного приложения.

В результате выполнения работы был получен опыт работы с операционной системой Android, языком программирования Kotlin, СУБД SQLite, а также получен опыт работы с многопоточностью и сопрограммами. Данное приложение дает возможность чтения электронных книг, добавления их в избранное, настройки оформления книги под свои нужды, а также прослушивания книг в аудиоформате.

Направления дальнейших исследований

Дальнейшие исследования и практические разработки будут направлены на расширение пользовательских возможностей приложения, а именно: возможность регистрации и авторизации пользователя, добавление возможности комментирования сохраненных цитат. Также планируется создание версии приложения для прослушивания книг в аудиоформате без использования интернета.

ЛИТЕРАТУРА

1. Словарь IT. [Электронный ресурс] URL: https://blog.skillfactory.ru/glossary/prilozhenie/ (дата обращения: 08.02.2024 г.).

2. В России растёт спрос на книги диджитал-авторов. [Электронный ресурс] URL: https://www.sostav.ru/publication/samizdat-65404.html (дата обращения: 08.02.2024 г.).

3. Словарь современного читателя. [Электронный ресурс] URL: https://eksmo.ru/slovar (дата обращения: 08.02.2024 г.).

4. Книги на «ЛитРес» озвучит искусственный интеллект. [Электронный ресурс] URL: https://godliteratury.ru/articles/2020/09/10/knigi-na-litres-ozvuchit-iskusstven (дата обращения: 08.02.2024 г.).

5. Букмейт приложение в Google Play. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.bookmate&pcampaignid=web_share (дата обращения: 08.02.2024 г.).

6. Литрес приложение в Google Play. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=ru.litres.android.global&pcampaignid=web_share (дата обращения: 08.02.2024 г.).

7. MyBook приложение в Google Play. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=ru.mybook&pcampaignid=web_share (дата обращения: 08.02.2024 г.).

8. Яндекс Старт приложение в Google Play. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=ru.yandex.search-plugin&pcampaignid=web_share (дата обращения: 08.02.2024 г.).

9. Зорина Н.В. Программирование на языке Джава: учебное пособие. // Москва: РТУ МИРЭА, 2021. – 164 с.

10. Герберт Ш. Java. Полное руководство. // Изд-во Диалектика Вильямс, 2020. – 629 с.

11. Kotlin для Android: теперь официально. [Электронный ресурс] URL: https://habr.com/ru/company/JetBrains/blog/329028 (дата обращения: 29.01.2024 г.).

12. Сравниваем Java и Kotlin. [Электронный ресурс] URL: https://habr.com/ru/company/otus/blog/580738 (дата обращения: 08.02.2024 г.).

13. IDE и редакторы кода для разработчиков. Подборка. [Электронный ресурс] URL: https://habr.com/ru/company/serverspace/blog/693374 (дата обращения: 08.02.2024 г.).

14. Словарь IT. [Электронный ресурс] URL: https://blog.skillfactory.ru/glossary/eclipse/ (дата обращения: 08.02.2024 г.).

15. Десять лучших IDE. [Электронный ресурс] URL: https://timeweb.com/ru/community/articles/5-luchshih-ide-1 (дата обращения: 08.02.2024 г.).

16. Android Studio: среда разработки мобильных приложений. [Электронный ресурс] URL: https://arduinoplus.ru/android-studio (дата обращения: 08.02.2024 г.).

17. Ботвинков А.В. Технологии проектирования информационных систем и технологий: учебное пособие. / А.В. Ботвинков, С.В. Моторин. // Новосибирск: СГУВТ, 2023. – 120 с.

18. Назарова О.Б. Моделирование бизнес-процессов: учебное пособие. / О.Б. Назарова, О.Е. Масленникова. // Москва: ФЛИНТА, 2023. – 261 с.

19. Синицын И.В. Разработка мобильных приложений: учебное пособие. / И.В. Синицын, Е.А. Чернов, Ю.А. Воронцов. // Москва: РТУ МИРЭА, 2023. – 162 с.

20. XML. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Web/XML/XML_introduction (дата обращения: 18.03.2024 г.).

21. Room. [Электронный ресурс] URL: https://developer.android.com/training/data-storage/room (дата обращения: 20.03.2024 г.).

22. Описание метода API v1. [Электронный ресурс] URL: https://yandex.cloud/ru/docs/speechkit/tts/request (дата обращения: 21.03.2024 г.).

23. Киреев Н.В. Разработка мобильных приложений на Котлин: учебное пособие. // Красноярск: СибГУ им. академика М. Ф. Решетнева, 2023. – 80 с.

24. Зубкова Т.М. Технология разработки программного обеспечения: учебное пособие. // Санкт-Петербург, 2022. – 324 с.

ПРИЛОЖЕНИЕ. Листинги программы

Листинг 1 – Разметка страницы главного экрана

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:an-
droid="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"

    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerViewEpubFiles"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <LinearLayout
        android:id="@+id/linearLayoutButtons"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@color/bottom_nav"
        android:padding="0dp"
        app:layout_constraintBottom_toBottomOf="parent">

        <ImageButton
            android:id="@+id/upload_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/upload_icon"
            android:background="?android:attr/selectableItemBackground"
            android:padding="16dp"
            android:layout_weight="1"/>

        <ImageButton
            android:id="@+id/filter_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/filter_icon"
            android:background="?android:attr/selectableItemBackground"
            android:padding="16dp"
            android:layout_weight="1"/>
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Листинг 2 – Разметка элемента списка книг

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="horizontal">
    <ImageView
        android:id="@+id/bookCoverImageView"
        android:layout_width="100dp"
        android:layout_height="150dp"
        android:scaleType="fitXY"
        android:src="@drawable/book_cover"
        />
```

```
        <TextView
            android:id="@+id/bookTitleTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="16sp"
            android:textColor="@color/black"
            android:maxLines="3"

    <ImageButton
        android:id="@+id/favouriteButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/heart_icon"/>

    <SeekBar
        android:id="@+id/readingSeekBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:progressDrawable="@drawable/custom_progress_bar"
        android:thumb="@null" />
</RelativeLayout>
```

Листинг 3 – Разметка страницы просмотра книги

```
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:an-
droid="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/coordinatorLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <WebView
        android:id="@+id/webView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

    <androidx.viewpager.widget.ViewPager
        android:id="@+id/viewPager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

    <LinearLayout
        android:id="@+id/bottomSheet"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_behavior="com.google.android.material.bottomsheet.Bot-
tomSheetBehavior">

        <TextView
            android:id="@+id/chapterNameTextView"
            android:layout_width="250dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="16dp"
            android:fontFamily="@font/avenirnextcyr_medium"
            android:layout_marginEnd="16dp"
            android:layout_marginTop="16dp"
            android:textSize="18sp"
            android:textColor="#000000"/>
```

Окончание листинга 3 приложения

```
        <SeekBar
            android:id="@+id/menuSeekBar"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:thumb="@drawable/custom_thumb"
            android:progressDrawable="@drawable/custom_seekbar"/>

         <ImageButton
            android:id="@+id/audioButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:src="@drawable/ic_audio"/>
</LinearLayout>
```